An Overview of the BlueGene/L Supercomputer
4/22/04

Abstract

This paper gives an overview of the BlueGene/L Supercomputer. This massively parallel system is based on a new scalable cellular architecture that exploits system-on-a-chip technology to deliver a peak processing power of up to 360 teraFLOPS (trillion floating-point operations per second). BlueGene/L systems can be assembled in a variety of configurations, starting with 512 compute nodes. The largest planned BlueGene/L machine will have 65,536 compute nodes and will be installed at the Lawrence Livermore National Laboratory. This machine is scheduled to be operational in the 2004-2005 time frame, at price/performance and power/performance levels that are unobtainable with current architectures.

1. Introduction and Background

IBM has previously announced a multi-year initiative to build a petaflop scale machine for calculations in the area of life sciences. . The BlueGene/L machine is the first step in this program, and is based on a different and more generalized architecture than IBM described in its announcement of the BlueGene program in December of 1999. In particular BlueGene/L is based on an embedded PowerPC processor supporting a large memory space, with standard compilers and a message passing enviroment, albeit with significant additions and modifications to the standard PowerPC system.

Significant progress has been made in recent years in mapping numerous compute-intensive applications (many of them grand challenges) to parallel architectures. This has been done to great success largely out of necessity, as it has become clear that currently the only way to achieve teraOPS-scale computing is to garner the multiplicative benefits offered by a massively parallel machine. To scale to the next level of parallelism, in which tens of thousands of processors are utilized, the traditional approach of clustering large, fast SMPs will be unsatisfactory. It is increasingly limited by power consumption and footprint constraints. For example, to house supercomputers in the 2004 time frame, both the Los Alamos National Laboratory and the Lawrence Livermore National Laboratory have begun constructing buildings with approximately 10x more power and cooling capacity and 2-4x more floor space than existing facilities. Due to the growing gap between the processor cycle times and memory access times, the fastest available processors will typically deliver a continuously decreasing fraction of their peak performance, despite ever more sophisticated memory hierarchies.

The approach taken in BlueGene/L (BG/L) is substantially different. This system is built out of a very large number of nodes, each of which has a relatively modest clock rate. These nodes present both low power consumption and low cost. The design point of BG/L utilizes IBM PowerPC embedded CMOS processors, embedded DRAM, and system-on-a-chip techniques that allow for integration of all system functions including the compute processor, communications processor, three levels of cache and multiple high speed interconnection networks with sophisticated routing on a single ASIC. With the relatively modest processor cycle time, the memory is close (in terms of cycles) to its processor. This is also advantageous for power consumption, and enables construction of denser packages in which 1024 compute nodes can be placed within a single rack. Integration of the inter-node communications network functions onto the same ASIC as the processors reduces cost since the need for a separate, high-speed switch is eliminated. The current design goals for BG/L aim for a scalable computer having up to 65,536 compute nodes and a target peak performance of 360 teraFLOPS with extremely cost effective characteristics and extremely low power (~1.6MW), novel cooling and low floor space (<2,500 sq ft) requirements. This peak performance metric is only applicable for applications that can utilize both of the processors available on a node for compute tasks. (We anticipate that there will be a large class of problems that will fully utilize one of the two processors in a node with messaging protocol tasks and will therefore not be able to utilize the second processor for computations.)

The BG/L design philosophy has been influenced by other successful massively parallel machines, including QCDSP at Columbia University. In that machine, thousands of processors are connected to form a multidimensional grid with nearest neighbor connections and simple global functions. Columbia University continues to evolve this architecture with their next generation QCDOC machine [QCDOC], which is being developed in cooperation with IBM research. QCDOC will also use a PowerPC processing core in an earlier technology, a simpler floating point unit, and a simpler nearest neighbor network.

2. System Overview

The BG/L supercomputer is designed to be the computational core of a larger system. BG/L itself is where parallel jobs from users run. It consists of 65,535 compute nodes and 1024 I/O nodes. Application processes, from the user jobs, execute on the compute nodes, under control of a light-weight operating system called the *Compute Node Kernel* (CNK). The I/O nodes run system services that support the execution of those application processes. Each I/O node runs an image of the Linux operating system. Each I/O node services 64 compute nodes. We call each group of one I/O node and its corresponding compute nodes a *processing set* (pset). BG/L is thus organized in 1024 psets. To assemble a complete system, we complement BG/L with a service node, front-end nodes, and file servers. For an illustration, see Figure 1.

The service node is where we run the *control system* of BG/L. The control system is responsible for the overall operation and health monitoring of the system. It consists of several components. The *Core Monitoring and Control System* (CMCS) is the component directly responsible for performing actions to the BG/L machine. CMCS performs these actions through a 1 Gbit/s Ethernet *control network* that connects the service node to approximately 2500 IDo chips. These IDo chips reside inside BG/L and are directly connected to all its nodes, power supplies, fans, temperature sensors, and other devices. The *Scheduler*.is responsible for machine partitioning and assignment of jobs to those partitions. The job scheduling services are responsible for receiving user job submission for execution, and then executing the job in the user identified partition The *Console* is the interface of the system administrator to the machine. Through the Console, the system administrator can monitor the status of BG/L and make configuration decisions. CMCS, the Console, and the Scheduler all use system *DB2 Database*. This database is the central repository of system state and also works as the communication medium for all the control system components. The service node for BG/L is implemented as a 16-processor pSeries machine, running the Linux operating system.

The front-end nodes are where the users of BG/L do their work. Users login to the front-end nodes to compile and build their applications. It is also in the front-end nodes that users submit their jobs to the Scheduler for execution. Finally, the user interface to debugging and performance analysis tools for BG/L also run in the front-end nodes. The front-end nodes of BG/L are implemented as a cluster of eight dual-processors pSeries machines, also running the Linux operating system.

Completing the entire system are the *file servers*. The file servers store the home directories of the users, containing the code and data for the applications that run in BG/L. They can also store checkpoint images from applications that run in BG/L. The file servers, the BG/L I/O nodes, the front-end nodes and the service node are all interconnected by a 1 Gbit/s Ethernet functional network. (Note that only the service node is connected to the control network.) This network is implemented by multiple switches, in order to provide the high I/O bandwidth required by the demanding applications expected to run in BG/L.

This combination of BG/L, as the computational core, with file servers, front-end nodes, and service node form a complete computing system that supports the execution of multiple high-performance applications, delivering both capability and capacity computing at unprecedented levels.

3. BG/L Compute Core Overview

The BlueGene/L compute core is a scalable subsystem in which the maximum number of compute nodes assigned to a single parallel job is $2^{16} = 65,536$. BlueGene/L is configured as a 64 x 32 x 32 three-dimensional torus of compute nodes. Each node consists of a single ASIC and memory. Each node can support up to 2 GB of local memory; our current plan calls for 9 SDRAM-DDR memory chips with 512 MB of memory per node. The ASIC that powers the nodes is based on IBM's system-on-a-chip technology and incorporates all of the functionality needed by BG/L. The nodes themselves are physically small, with an expected 11.1-mm square die size, allowing for a very high density of processing. The ASIC uses IBM CMOS CU-11 0.13 micron technology and is designed to operate at a target speed of 700 MHz, although the actual clock rate used in BG/L will not be known until chips are available in quantity.

The design for the BG/L system packaging is shown in Figure 22. The design calls for 2 nodes per compute card, 16 compute cards per node board, 8 node boards per 512-node midplane of approximate size 17"x 24"x 34," and two midplanes in a 1024-node rack. Each processor can conduct 4 floating point operations per cycle (in the form of two floating point multiply-add's per cycle); at the target frequency this amounts to approximately 1.4 teraFLOPS peak performance for a single midplane of BG/L nodes, if we count only a single processor per node. Each node contains a second processor, identical to the first, although not included in the 1.4 teraFLOPS performance number which is intended primarily for handling message passing operations.

The system also provides for a flexible number of dual-processor "I/O nodes". They can be configured with processor node groups in ratios of 1:8 to 1:64 (I/O Node : Compute Node). Each "I/O nodes" uses the same ASIC as the compute nodes, also with external memory and active gigabit Ethernet connections.

Each compute node executes a lightweight kernel. The compute node kernel handles basic communication tasks and all the functions necessary for high performance scientific code.

A host computer is required for compiling, diagnostic support and problem analysis. I/O nodes handle the communications between the compute nodes and other systems, including the host and the file servers. The choice of host will depend on the class of applications exercised and their bandwidth and performance requirements.

The nodes are interconnected through five networks: a 3D torus network for point-point messaging between compute nodes, a global combining/broadcast tree for collective operations such as MPI_Allreduce over the entire application, a global barrier and interrupt network, a Gigabit Ethernet to JTAG network for machine control, and another Gigabit Ethernet network for connection to other systems, such as hosts and file systems. For cost and overall system efficiency, compute nodes are not hooked directly up to the Gigabit Ethernet, but rather use the global tree for communicating with their I/O nodes, while the I/O nodes use the Gigabit Ethernet to communicate to other systems.

In addition to the compute ASIC, there is a "link" ASIC. When crossing a midplane boundary, BG/L's torus, global combining tree and global interrupt signals pass through the BG/L link ASIC. This ASIC serves two functions. First, it redrives signals over the cables between BG/L midplanes, improving the high-speed signal shape and amplitude in the middle of a long, lossy trace-cable-trace connection between nodes on different midplanes. Second, the link ASIC can redirect signals between its different ports. This redirection function enables BG/L to be partitioned into multiple, logically separate systems in which there is no traffic interference between systems. This capability also enables additional midplanes to be cabled as spares to the system and used, as needed, upon failures. Each of the partitions formed through this manner has its own torus, tree and barrier networks which are isolated from all traffic from all other partitions on these networks.

System fault tolerance is a critical aspect the BlueGene/L machine. BlueGene/L has many layers of fault tolerance that are expected to allow for good availability despite the large number of system nodes. In addition, the BlueGene/L platform will be used to investigate many avenues in autonomic computing.

4. System Packaging

The BG/L system is a cost/performance design, focused on fault tolerance, high density, low power and thus achieving low acquisition and runtime cost. The hardware cost is dominated by the ASIC and DRAM devices themselves. To manage circuit card costs, the interconnect was developed from the outside in. After identifying a system package based on standard 19" racks modified for high power transverse air cooling, the racks were arranged to minimize the longest cable as this is the bandwidth limiter in rack to rack communication. The differential cables are 26 AWG, and the longest length is 8 meters. (Thicker conductors gave negligible improvement in attenuation for the added bulk.) The layout focus was on mechanical footprint, robustness and avoidance of large impedance discontinuities. Connector pin assignments for cables and circuit cards were chosen to minimize card wiring layers and avoid high priced, higher risk fine geometries, while also minimizing the worse case Manhattan wire lengths. Circuit cards and ASIC packages have just 4 internal wiring layers except for the 6 wiring layer midplane. Minimum card line width and space is ~110u, long traces are oversized to ~200u width while maintaining 50Ω impedance to reduce DC & skin effect losses. ASIC floorplanning and I/O assignments were iterated until a minimum layer ASIC package could be built. To avoid large numbers of capacitors for proper image current return at connectors, and to reduce switching noise, all differential signaling was used for the 1.4 Gb/s torus and tree links, and complete ground referencing on all nets. To reduce connector failure all DRAMs and DC-DC power supplies are directly soldered, cables connectors have screwed lockdowns, and all connectors are extremely reliable, pin and socket multi-contact interfaces. Each 22 differential pair cable contains a spare pair which can be swapped in by the link ASIC, in much the same way that the 9 chip DRAM system has a spare 4 bits (nibble) that can be swapped in if required. The DRAM additionally supports through ECC (error correcting code) the ability to run uninterrupted without error when losing a consecutive byte of the data bus as well as the usual single bit correct/ double bit detect functionality provided by ECC. Cables are removed only to service the link cards.

The BlueGene/L power and cooling network is an example of a cost/performance fault tolerant design. The system is air cooled, designed to operate in standard raised floor machine rooms, and assumes standard fault tolerant 220V rack feed and failover air chillers. Racks are designed to be on 3 ft pitch in a row and 6 ft pitch between rows, so required perforations in 2 ft sq floor tiles are replicable. Chilled air to cool the expected 25KW/rack is drawn from an opening in the raised floor beneath the rack by a wall of fans on the left side of the rack. As shown in Figure 3, thirty 100mm diameter high speed, DC "smart fans" arranged on pluggable fans cards cool a midplane. Fan speed is monitored and adjusted with feedback from thermal sensors to maintain constant chip temp in the face of coherent system-wide power demands, this reduces the effect of mini-cycles which can cause thermal cycle fatigue of chip to card connections. If a fan slows or stops, the others increase rotation to maintain ASIC junction temperature and an error condition is reported to the control host. A damaged fan card can be replaced with the system running. Fan power is supplied by the same 208V AC->48V DC N+1 redundant supplies that power the rack electronics. The 48V DC is further regulated to the 1.5V and 2.5V required respectively by the BG/L ASICs and external DRAMs by either commercially available long-lived converters with a mean time between failures (MTBF) of 2 Million hours, or redundant supplies. Both design points are being considered, the desired MTBF of the system is at least 10 days.

The system MTBF is calculated below assuming predicted failure rates for ASICs after burn-in, predicted DRAM hard failure rates, and manufacturer's suggested average failure rates for remaining components. Redundant power supplies would further increase MTBF. The MTBF is expected to be dominated by DRAM hard failures. The expected DRAM failure rate is ~5x less than the raw DRAM rate of 25FITs when the effects of bit sparing in the controller are accounted for. The maintenance policy will track soft errors and replaces nodes at service intervals when DRAMs or compute ASICs with EDRAM show increased soft error rates. This would further reduce the MTBF value.

System design methodology includes extensive use of parity and ECC to allow for the detection (and possible correction) for the vast majority of soft error events.

Table 1:Uncorrectable Hard Failure Rates of BlueGene/L by major component

Component	FIT per component*	Components per 64k partition	FITs per system	Failure rate per week
Ethernet complex	160	3024	484k	
DRAM	5	608,256	3,041k	
Compute + I/O ASIC	20	66,560	1,331k	
Link ASIC	25	3072	77k	
Clock chip	6.5	~1200	8k	
Non-redundant power supply	500	384	384k	
Total (65,536 compute nodes)			5315k	0.89

* After burn-in and applied redundancy. T=60C, V=Nom, 40K POH. FIT = Failures in ppm/KPOH. 1 FIT = 0.168*10^-6 fails/wk if machine runs 24 hrs/day.

5. Node Overview

The BG/L node ASIC, shown in Figure 4, includes two standard PowerPC 440 processing cores, each with a PowerPC 440 FP2 core. Thjs is the enhanced "Double" 64-bit Floating-Point Unit. The 440 is a standard 32-bit microprocessor core from IBM's Microelectronics division. This superscalar core is typically used as an embedded processor for both internal and external customer applications. Since the 440 CPU core does not implement the necessary hardware to provide SMP support, the two cores are not L1 cache coherent. A lockbox is provided to allow coherent processor-to-processor communication. The chip also includes a very small L2 cache which is used as a prefetch buffer, a fast SRAM array for communication between the two cores, an L3 cache directory and 4MB of associated L3 cache made from embedded DRAM, integrated external DDR memory controller, gigabit Ethernet adapter, a JTAG interface as well as all the network link cut-through buffers and control. The L2 and L3 are coherent between the two cores.

In normal operating mode, one CPU/FPU pair is used for computation while the other is used for messaging. However, there are no hardware impediments to fully utilizing the second processing element for algorithms that have simple message passing requirements (i.e., those with a large compute to communication ratio).

The PowerPC 440 FP2 core, shown in Figure 5, consists of a primary side and a secondary side, each of which is essentially a complete floating-point unit. Each side has its own 64-bit by 32 element register file, a double-precision computational data path and a double-precision storage access data path. A single common interface to the host PPC 440 processor is shared between the sides.

The primary side is capable of executing standard PowerPC floating-point instructions, and acts as an off-the-shelf PPC 440 FPU. An enhanced set of instructions include those that are executed solely on the secondary side, and those that are simultaneously executed on both sides. While this enhanced set includes SIMD operations, it goes well beyond the capabilities of traditional SIMD architectures. Here, a single instruction can initiate a different yet related operation on different data, in each of the two sides. These operations are performed in lockstep with each other. These types of instructions have been termed SIMOMD for Single Instruction Multiple Operation Multiple Data. While Very Long Instruction Word (VLIW) processors can provide similar capability, this FPU is able to provide it using a short (32 bit) instruction word, avoiding the complexity and required bandwidth of long instruction words.

Another advantage over typical SIMD architectures is the ability of either of the sides to access data from the other side's register file. While this saves a lot of swapping when working purely on real data, its greatest value is how it simplifies and speeds up complex-arithmetic operations. Complex data pairs can be stored at the same register address in the two register files with the real portion residing in the primary register file, and the imaginary portion residing in the secondary register file. Newly defined complex-arithmetic instructions take advantage of this data organization.

A quadword (i.e., 128 bits) data path between the PPC 440s Data Cache and the PPC 440 FP2 allows for dual data elements (either double-precision or single precision) to be loaded or stored each cycle. The load and store instructions allow primary and secondary data elements to be transposed, speeding up matrix manipulations. While these high bandwidth, low latency instructions were designed to quickly source or sink data for floating-point operations, they can also be used by the system as a high speed means for transferring data between memory locations. This can be especially valuable to the message processor.

The PowerPC 440 FP2 is a superscaler design supporting the issuance of a computational type instruction in parallel with a load or store instruction. Since a fused multiply-add type instruction initiates two operations (i.e., a multiply and an add or subtract) on each side, four floating-point operations can begin each cycle. To help sustain these operations, a dual operand memory access can be initiated in parallel each cycle.

The core supports single element load and store instructions such that any element, in either the primary or secondary register file, can be individually accessed. This feature is very useful when data structures in code (and hence in memory) do not pair operands as they are in the register files. Without it, data might have to be reorganized before being moved into the register files, wasting valuable cycles.

Data is stored internally in double-precision format; any single-precision data is automatically converted to double-precision format when it is loaded. Likewise, when data is stored via a single-precision operation, it is converted from double to single precision, with the mantissa being truncated as necessary. In the newly defined instructions, if the double-precision source is too large to be represented as a single-precision value, the returned value is forced to a properly signed infinity. However, round to single precision instructions are provided so that an overflowing value can be forced to infinity or the largest single precision magnitude, based on the rounding mode. Furthermore, these instructions allow for rounding of the mantissa.

All floating-point calculations are performed internally in double precision and are rounded in accordance with the mode specified in the PowerPC defined floating-point status and control register (FPSCR). The newly defined instructions produce the IEEE-754 specified default results for all exceptions. Additionally, a non-IEEE mode is provided for when it is acceptable to flush denormalized results to zero. This mode is enabled via the FPSCR and it saves the need to renormalize tiny results when using them as inputs to subsequent calculations.

All computational instructions, except for divide and those operating on denormalized operands, execute with a five cycle latency and single cycle throughput. Division is iterative, producing two quotient bits per cycle. Division iterations cease when a sufficient number of bits are generated for the target precision, or the remainder is zero. Faster division can be achieved by employing the highly accurate (i.e., to one part in 2^{13}) reciprocal estimate instructions and performing software-pipelined Newton-Raphson iterations. Similar instructions are also provided for reciprocal square root estimates with the same degree of accuracy.

Since the instruction set was extended beyond the PowerPC architecture, the necessary compiler enhancements are being developed. Library routines and ambitious users can also exploit these enhanced instructions through assembly language, compiler builtin functions, and advanced compiler optimization flags. The double FPU can also be used to advantage by the communications processor, since it permits high bandwidth access to and from the network hardware.

Power is a key issue in such large scale computers, therefore the FPUs and CPUs are designed for low power consumption. Incorporated techniques range from the use of transistors with low leakage current, to local clock gating, to the ability to put the FPU or CPU/FPU pair to sleep. Furthermore, idle computational units are isolated from changing data so as to avoid unnecessary toggling.

The memory system is being designed for high bandwidth, low latency memory and cache accesses. An L2 hit returns in approximately 10 processor cycles, an L3 hit in about 25 cycles, and an L3 miss in about 75 cycles. L3 misses are serviced by external memory, the system in design has a 16 byte interface to nine 512Mb SDRAM-DDR devices operating at a speed of ½ of the processor. While peak memory bandwidths are high, sustained bandwidths will be lower for certain access patterns, such as a sequence of loads, since the 440 core only permits three outstanding loads at a time.

The high level of integration of the BlueGene/L system-on-a-chip approach allows for latencies and bandwidths that are significantly better than those for nodes typically used in ASCI scale supercomputers.

6. Torus Network

The torus network is used for general-purpose, point-to-point message passing and multicast operations to a selected "class" of nodes. The topology is a three-dimensional torus constructed with point-to-point, serial links between routers embedded within the BlueGene/L ASICs. Therefore, each ASIC has six nearest-neighbor connections, some of which may traverse relatively long cables. The target hardware bandwidth for each torus link is 175MB/sec in each direction.

The general structure of the torus within each node is shown in Figure 6. Packets are injected into the network at one of the Local Injection FIFOs, and are deposited into a Local Reception FIFO upon reaching their destinations. The messaging co-processor is responsible for injecting and removing packets to and from these FIFOs. Packets that arrive from another node are immediately forwarded in the absence of contention, or stored in a waiting FIFO in the corresponding input unit until the contention clears. Arbitration is highly pipelined and distributed (each input and output unit has its own arbiter), as is common in such switches. Using a link-level CRC and a HIPPI-like retransmission protocol, the network will reliably deliver a single copy of every packet injected.

The torus network provides both adaptive and deterministic minimal-path routing, and is deadlock free. Throughput and hardware latency are optimized through the use of virtual cut-through (VCT) routing. Messages can be composed of multiple packets, which are the atomic unit of routing. Therefore, adaptively routed packets from the same message can arrive out of order. Packets are variable in size, ranging from 32 bytes to 256 bytes with a granularity of 32 bytes.

Virtual channels (VCs) are used to provide deadlock-free adaptive routing and increase throughput. A "near cycle accurate" simulator of the torus network has been developed that has been used for the detailed design of the network. Based on performance studies and a sizing of the hardware requirements, the network will have four VCs. Two VCs will be devoted to adaptive routing, and two will be devoted to deterministic routing. One of the deterministic VCs is used as a "bubble escape channel" for the adaptive sub-network in order to guarantee deadlock freedom, and the other is reserved for high-priority packets. Because it is expected that most traffic will be adaptively routed, two adaptive VCs are provided in order to reduce head-of-line blocking and allow for the use of simple FIFO buffers within the routers. Flow control between routers is provided through the use of tokens. There is sufficient buffer space to maintain full link hardware bandwidth in the absence of contention.

One of the most communications intensive operations in scientific computing is the MPI_Alltoall collective communications call in which every processor sends a different message to every other processor. We simulated a representative portion of this call on a 32K node machine and the results are displayed in Table 2. The table illustrates the advantage of dynamic routing over static routing and shows that, because of the highly pipelined nature of the network, the links can be kept busy essentially all the time with only 2 dynamic VCs. For full sized packets, we expect a 12% overhead due to the hardware packet header and trailer, the software packet header, and the token protocol; this accounts for the difference between the total link utilization and the payload link utilization.

	Average Link Utilization (Total)	Average Link Utilization (Payload Only)
Static Routing	76%	66%
1 Dynamic VC	95%	83%
2 Dynamic VCs	99%	87%

Table 2: Estimated link utilizations during the middle of an MPI_Alltoall operation on a 32K node BG/L. In all three cases, the total FIFO sizes are fixed and equal to 3 KB per link. There is no high priority traffic in this exchange.

7. Signaling

The BG/L torus interconnect, and the BG/L tree described below, rely on serial communication. A common system wide clock at the frequency of the processor is used to provide a reference for sending and receiving data. Data is driven on both clock edges from a differential, two bit pre-compensating driver designed to overcome the ~10 decibel of loss on the 8 meter differential cables and connectors. Data is captured by over-sampling using a string of time delayed latches, the location of the data bit is computed by a background state machine that monitors false transitions and tracks changes in arrival or sampling times. To reduce power and allow for minimum silicon delay, a variable delay line after the driver is auto-configured at power-on to optimize the location of the sampled datum in the instrumented delay line. Features such as byte serial/deserializing, parity and CRC generation and checking, message retry, and checksums for error localization are all provided by the hardware.

8. Global Trees

Message passing on the global combining tree is done through the use of a packet structure similar to that of the torus network. The tree network is a token-based network with two VCs. Packets are non-blocking across VCs. Setting programmable control registers flexibly controls the operation of the tree. In its simplest form, packets going up towards the root of the tree can be either point-to-point or combining. Point-to-point packets are used, for example, when a compute node needs to communicate with its I/O node. The combining packets are used to support MPI collective operations, such as MPI_Allreduce, across all the nodes connected to the tree (e.g., on MPI_COMM_WORLD). All packets coming down the tree are broadcast further down the tree according to the control registers and received upon reaching their destination. For collective operations, the packet is received at each node. The tree has a target hardware bandwidth of 350 MB/sec and a target one-way hardware latency of about 2.5 microseconds on a 64K node partition.

The hardware associated with the tree is all contained in the ASIC. The tree handles basic integer arithmetic functions such as max and sum (on integers of certain selectable widths), and bit-wise operations such as xor. To do a floating point sum reduction on the tree requires potentially two round trips on the tree. In the first trip each processor submits the exponents for a max reduction. In the second trip, the mantissas are appropriately shifted to correspond to the common exponent as computed on the first trip and then fed into the tree for an integer sum reduction.

A separate set of wires based on asynchronous logic form another tree that enables fast signaling of global interrupts and barriers (global AND or OR). The target latency to perform a global barrier over this network for a 64K node partition is approximately 1.5 microseconds.

9. BlueGene/L System Software

Scalable system software that supports efficient execution of parallel applications is an integral part of the BlueGene/L architecture. A BlueGene/L application is organized as a collection of compute processes, each executing on its own compute node. I/O nodes provide additional services for the executing application. The machine can be partitioned for the execution of multiple applications, with each application running on its own dedicated partition. This section describes the various components of the BlueGene/L system software stack. We cover basic compute and I/O node software, compilers and application development tools, computation and communication libraries, recovery features, and system management and control software. We also discuss the parallel programming models that are being pursued for BlueGene/L.

9.1 Basic Compute and I/O Node Software

The goal in developing the system software for BG/L has been to create an environment which feels familiar to the application programmer and also delivers high levels of application performance. Users develop, debug, and submit their jobs for execution from front-end nodes, which are commercially available Linux machines (see Section 2). Application processes always execute in the compute nodes, which implement a Unix-like environment for those processes. The approach adopted in implementing the execution environment for user processes was to split the operating system functionality between compute and I/O nodes.

The compute node operating system, called the BlueGene/L *Compute Node Kernel* (CNK), is a simple, lightweight, single-user operating system that operates in one of two modes, selected at boot time: *coprocessor mode* and *virtual node mode*. In coprocessor, mode, CNK supports execution of a single dual-threaded application compute process.

Each thread of the compute process is bound to one of the processors in the compute node. In virtual node mode, CNK supports the execution of two single-threaded processes. Each process is bound to one of the processors in the compute node.

BlueGene/L applications are linked with a set of user-level runtime libraries that interface to the CNK to implement the services normally expected by Unix processes (file and socket I/O, math operations, string manipulation, dynamic memory allocation, etc). Applications are also linked with a BlueGene/L-specific runtime library that provides the compute process with direct access to the torus and tree networks. Together, CNK and this runtime library implement compute node-to-compute node communication through the torus and compute node-to-I/O node communication through the tree. The compute node-to-compute node communication is intended for exchange of data by the application. Compute node-to-I/O node communication is used primarily for extending the compute process into an I/O node, so that it can perform services available only in that node.

Application processes running on compute nodes also have access to the extensive performance monitoring hardware in BlueGene/L. A compute node has a set of 50 performance counters that can be programmed to count a large variety of events, from floating-point instructions to network transmissions. These performance counters are accessible to the running process through the PAPI standard.

The lightweight kernel approach for the compute node was motivated by the experience with the Puma and Cougar kernels at Sandia National Laboratory and University of New Mexico. The BlueGene/L compute kernel provides a single and static virtual address space to one running compute process. (Two processes in virtual node mode.) Because of its single-process nature, the BlueGene/L compute kernel does not need to implement any context switching. It does not support demand paging and exploits large pages to ensure complete TLB coverage for the application's address space. This approach results in the application process receiving full resource utilization. There is always a physical processor supporting the execution of a user-level thread.

The I/O nodes run the Linux operating system, which supports the execution of multiple processes in those nodes. Only system software executes on the I/O nodes, no application code. The purpose of the I/O nodes during application execution is to complement the compute nodes with services that are not provided by the compute node software. I/O nodes provide an actual file system to the running applications. They also provide streaming connections to processes in other systems. The I/O node also performs process authentication, accounting, and authorization on behalf of its compute nodes.

The interaction between compute and I/O nodes happens through a process we call *function shipping*. When a compute process in a compute node performs an I/O operation (on a file or a stream), that I/O operation (e.g., a read or a write) is shipped through the tree network to a service process in the I/O node. That service process then issues the operation against the I/O node operating system. The results of the operation (e.g., return code in case of a write, actual data in case of a read) are shipped back to the originating compute node. This separation of services between compute and I/O nodes enables us to keep the compute node software simple (thus increasing its reliability) and free of asynchronous events (thus improving system performance).

I/O nodes also provide a debugging interface for user applications. Debuggers running on the front-end nodes use I/O node services to debug application processes running on compute nodes. In this case, the shipping occurs in the opposite direction. Debugging operations performed on the front-end node to the I/O node are shipped to the compute node for execution against a compute process. Results are shipped back to the debugger through the I/O node.

9.2 Compilers and Application Development Tools

As we previously discussed, all application development for BlueGene/L is done on conventional front-end nodes. IBM provides C, C++ and Fortran compilers from its XL family for BlueGene/L. The compilers include advanced loop transformations and inter-procedural analysis for optimizing applications. The XL compilers for BlueGene/L have been enhanced with code generation and optimization for the double floating-point unit in the BlueGene/L compute nodes.

Performance data is gathered by running processes using the services in the compute nodes. This data can be both from performance counters, obtained through PAPI, and from instrumentations inserted in the MPI message-passing library. Performance analysis can then be done with a variety of tools in the front-end nodes. The MPICH2 software that we use in BlueGene/L includes its own set of visualization tools. In addition to that, we are working with Universitat Polytecnica de Catalunya to port their ParaVer and Dimemas tools to BlueGene/L.

Parallel debuggers represent another component of the BlueGene/L application development environment. IBM is working with Etnus to deploy TotalView on BlueGene/L. Users will interact with the TotalView user interface, also called the TotalView *client*, running on front-end nodes. The client in turn communicates with multiple instances of TotalView *servers*, which run on each I/O node. The TotalView server uses services in the I/O node to debug application processes running on the compute nodes.

9.3 Communication Libraries

MPI is the primary communication library for BlueGene/L applications. In fact, it is fair to say that BlueGene/L is designed and optimized to run large MPI programs. IBM has worked with Argonne National Laboratory to implement a highly optimized MPI library for BlueGene/L using MPICH2. MPI is implemented in BlueGene/L entirely in user mode. Because there is a single job per partition and a single process per node (physical or virtual), the communication hardware can be exposed directly to user mode without any security issues.

MPICH2 for BlueGene/L makes effective use of the various interconnection networks of BlueGene/L. The torus is used to implement the basic point-to-point operations, as well as the generic versions of all the collectives. The tree is used to implement optimized versions of reduce and broadcast operations on the MPI_COMM_WORLD (and duplicates) communicators. The global barrier network is used to implement fast barriers across an entire partition. MPICH2 makes full use of the two processors in a BlueGene/L compute node. In coprocessor mode, the second processor is used as a message-passing engine to improve the MPI communication bandwidth. In virtual node mode, each processor of a node runs a separate MPI task and implements the full MPI functionality.

MPICH2 for BlueGene/L will implement the entire MPI-2 standard, except for dynamic process management. Deployment of this functionality will be in stages. We expect to have full MPI 1.2 standard support by the end of 2004. We will also provide basic MPI-I/O functionality by end of 2004, although performance optimizations will only be available in mid-2005. BlueGene/L's MPI-I/O will support the porting of important I/O libraries used by many applications, including HDF5 and PNetCDF.

MPI-2 one-sided communications will itself be deployed in stages. In the first phase, one-sided communications with *active targets* (i.e., targets that actively performing MPI operations) will be deployed by mid-2005. In a second phase, we will support one-sided communication involving *passive targets* (i.e., targets that are not necessarily doing any MPI operations).

9.4 Computation Libraries

Computation libraries represent an effective way to deliver efficient use of BlueGene/L performance features to user applications. IBM is working to port a variety of mathematical libraries to BlueGene/L, including ESSL, MASS, and MASSV. ESSL (Engineering and Scientific Subroutine Library) is a collection of math functions for matrix and vector operations, including matrix factorization and linear system solving functions. MASS (Math Acceleration SubSystem) is a collection of elementary functions like sin, cos, and log. MASSV (Math Acceleration SubSystem Vector) implements the vector form of those elementary functions. All three libraries are optimized to take advantage of the dual floating-point units in each BlueGene/L compute node.

An important part of the IBM strategy to deliver high-performance computation libraries to BlueGene/L is to work with collaborators. We are already working with the Technical University of Vienna to develop an optimized FFT library. We also want to pursue the porting of higher level libraries, like PETSc and Overture, to BlueGene/L.

9.5 Recovery Features

.
Given the scale of BG/L, there is clearly a need to recover from failures of individual components. Support for long-running applications will be provided through a checkpoint/restart mechanism. An application-assisted checkpoint infrastructure is currently in development. In this approach, the application programmer is responsible for

identifying points in the application in which there are no outstanding messages. The programmer can then place calls to a system checkpoint in those points. When executed, the checkpoint service will synchronize all tasks of the application and take a complete application checkpoint, writing the state of all compute processes to disk. Application state that resides on I/O nodes, particularly file pointers and list of open files, is also saved to disk. In case of unexpected termination, the application can then be restarted from its latest checkpoint.

9.6 System Management and Control Software

The BG/L system software includes a set of control services that execute on the service node. Many of these services, including system bringup, machine partitioning, system performance measurements, and monitoring of system health, are not architected from a user perspective. They are performed through the control network under commands from a job scheduling entity and/or the system administrator.

Job scheduling and job start are two services implemented by the control software that are visible to end users. Job scheduling in BlueGene/L is performed by LoadLeveler. Using standard LoadLeveler in BlueGene/L supports easy interoperability with other IBM systems and also leverages Grid support that is being incorporated into that product. BlueGene/L can execute both batch and interactive jobs. Batch jobs are submitted to LoadLeveler using conventional job command files. When resources are allocated, LoadLeveler executes the user job script in a front-end node. That script typically contains one or more mpirun commands to initiate execution of a parallel job in the BlueGene/L computational core. Alternative, a user can run a job interactively by executing an mpirun command directly on a front-end node. In that case, mpirun contacts LoadLeveler to obtain and allocation and then starts the parallel job. In BlueGene/L, mpirun starts a job by invoking services in the service node.
The BlueGene/L control system is centered on a DB2 database that resides in the service node. That database is a central repository for four different types of data. Configuration data describes the physical hardware of the machine, including all the nodes and cables that are present. Operational data describes how a machine is partitioned and which job is running on each partition. Job schedulers and system consoles act on the system by manipulating this operational data. Environmental data consists of measurements of physical values of the system, like chip temperatures and fan speeds. RAS (Reliability, Serviceability, and Availability) consists of a log of the events (typically errors) that were reported by the system software running on I/O and compute nodes.

9.7 Programming Models

Message passing with MPI is expected to be the dominant parallel programming model for BG/L applications. Nevertheless, IBM is working with several collaborators to deploy other programming models for BlueGene/L. We are working with Pacific Northwest National Laboratory to implement RDMA services that could be used to support Global Arrays and Co-Array Fortran in BlueGene/L. We are also working with the University of Illinois to implement Charm++ and Hierarchically Tiled Arrays, in support of applications which have a more irregular structure. Charm++ has already been demonstrated running on BlueGene/L. UPC (Unified Parallel C) is another programming model that is being pursued for BlueGene/L.

10. Validating the architecture with application programs

There are a large number of applications that have been ported to the BlueGene/L architecture. It is difficult to generalize which scientific fields of study are appropriate for this machine but it is possible to give some guidance as to what type of algorithms are well suited to the BlueGene/L machine architecture.

The types of algorithms that tend to be well suited to the BlueGene/L machine architecture have the following properties.

- Algorithm is inherently parallel. In general seeing good scaling of the algorithm on any other massively parallel architecture is sufficient to show this.
- The memory footprint per MPI task is less than 500MB. This is the amount of memory available on a standard BlueGene/L node. Other memory size options are being considered.
- The code is written using message passing (MPI). This is currently the only messaging supported in the BlueGene/L machine. The MPI library has been optimized for the BlueGene/L architecture.
- If the code is dominated by collective communication, then these communications are best done on all nodes (MPI_COMM_WORLD). While all communicators are supported, collectives on this communicator (and duplicates of this communicator) are optimized to utilize the tree network efficiently and will see marked performance enhancement. Other communicators are supported and will use the torus network.
- Code that has frequent short vector or scalar global reductions will see excellent performance on BlueGene/L when compared to other architectures. This is a result of the tree network latency being orders of magnitude shorter than the latency of other supercomputers.
- If file accesses are infrequent and the application is not file input/output dominated on other massively parallel machines then file I/O is probably not a problem.
- Applications which when mapped onto a three-dimensional grid have communications that are usually local in this three-dimensional mapping will see excellent network performance. This is typical of grid-decomposed problems with short range interactions.

There are also some general guidelines to the type of algorithms that achieve good single node performance. BlueGene/L has a standard multi-cache level memory system with excellent bandwidths and latencies so in general good performance is seen from the memory system.

- Algorithms that utilize vector floating point operations will often get good use of our double floating point unit thereby getting a good speed-up.

- Algorithms that have very poor cache hit rates will usually see significantly better performance on BlueGene/L than on other machines. This is a relative statement as extremely poor cache hit rates will almost always result in sustained performance being a small fraction of peak performance.

There are algorithms that will have difficulty on the BlueGene/L architecture but in general these have difficulties on all parallel architectures. Algorithm features that will result in difficult scaling are:

- Codes that are network communication dominated with small messages are typically difficult. In fact BlueGene/L will often do better than other machines due its ability to support very small packet and message sizes.
- Codes that are communication bound and communicate between nodes that have no locality when mapped onto a three-dimensional grid will show difficulties in BlueGene/L.
- Codes that have a master/slave structure will be difficult to scale to large node counts because of the usual Amdahl's law limitation coupled with modest single node performance.

11. Conclusions

BlueGene/L represents a new and exciting development in high-performance computing. Its cellular architecture approach, supported by multiple fast interconnection networks, supports almost limitless scalability. The BlueGene/L installation at Lawrence Livermore National Laboratory will be the first machine in the world to break both the 100,000 processor barrier and the 100 Tflops barrier. When the full 65,536-compute node system is installed in 2005, it will deliver more than 360 Tflops, almost 10 times the peak performance of today's fastest machine. The same architecture also supports machines with 1,000 processors and close to 3 Tflops of peak performance.

This level of parallelism and performance will enable a level of computing that is just not possible today. With it, new scientific discoveries and technological advances will become reality. We have just started exploring the spectrum of applications for BlueGene/L, but there is no doubt that its impact will be felt across the high performance computing community.

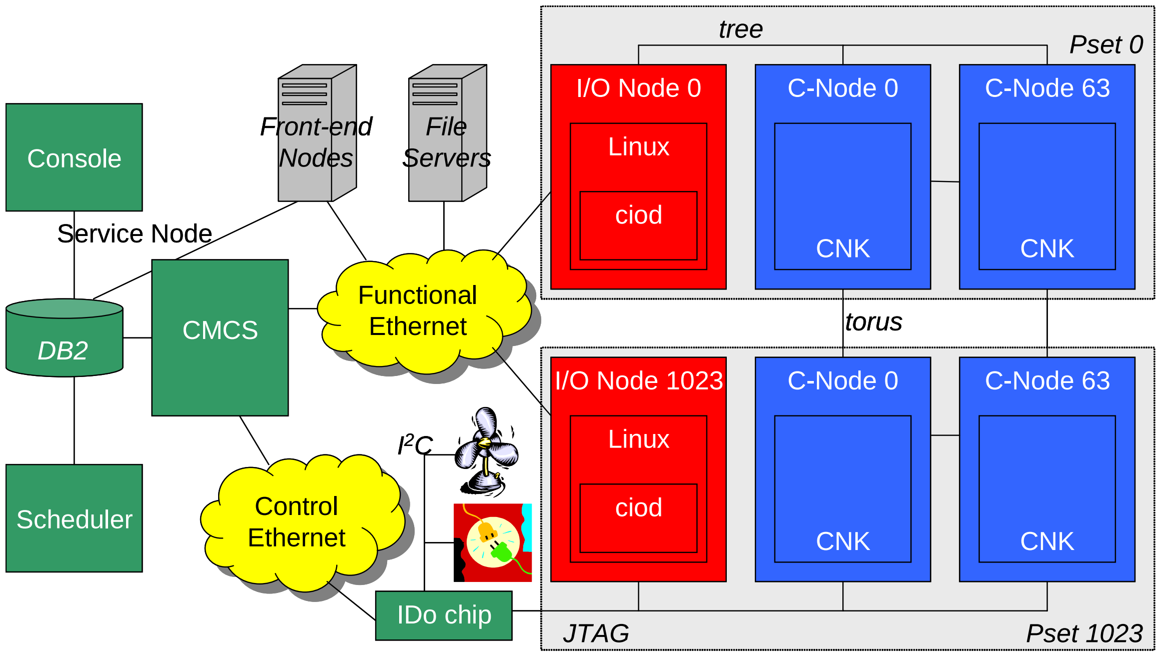

Figure 1: Overview of a complete system with BlueGene/L as the computational core.

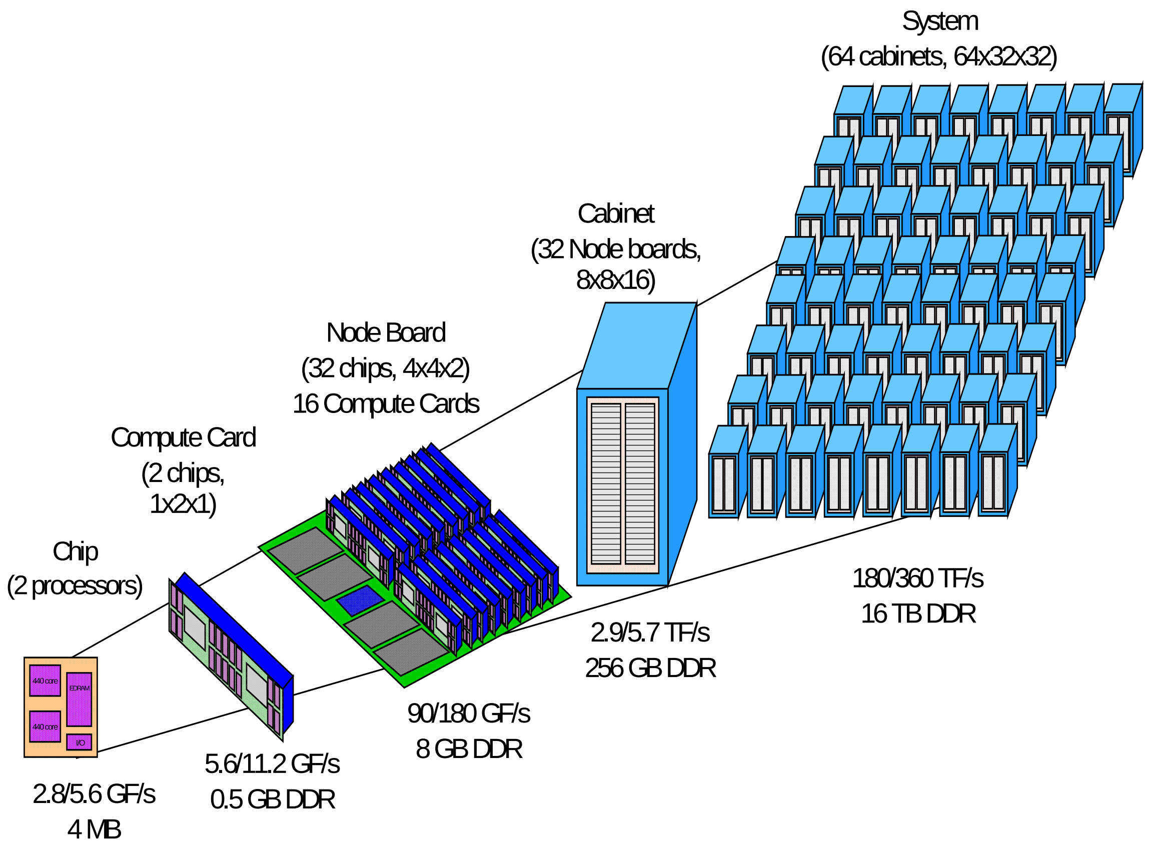

Figure 2: BlueGene/L Packaging

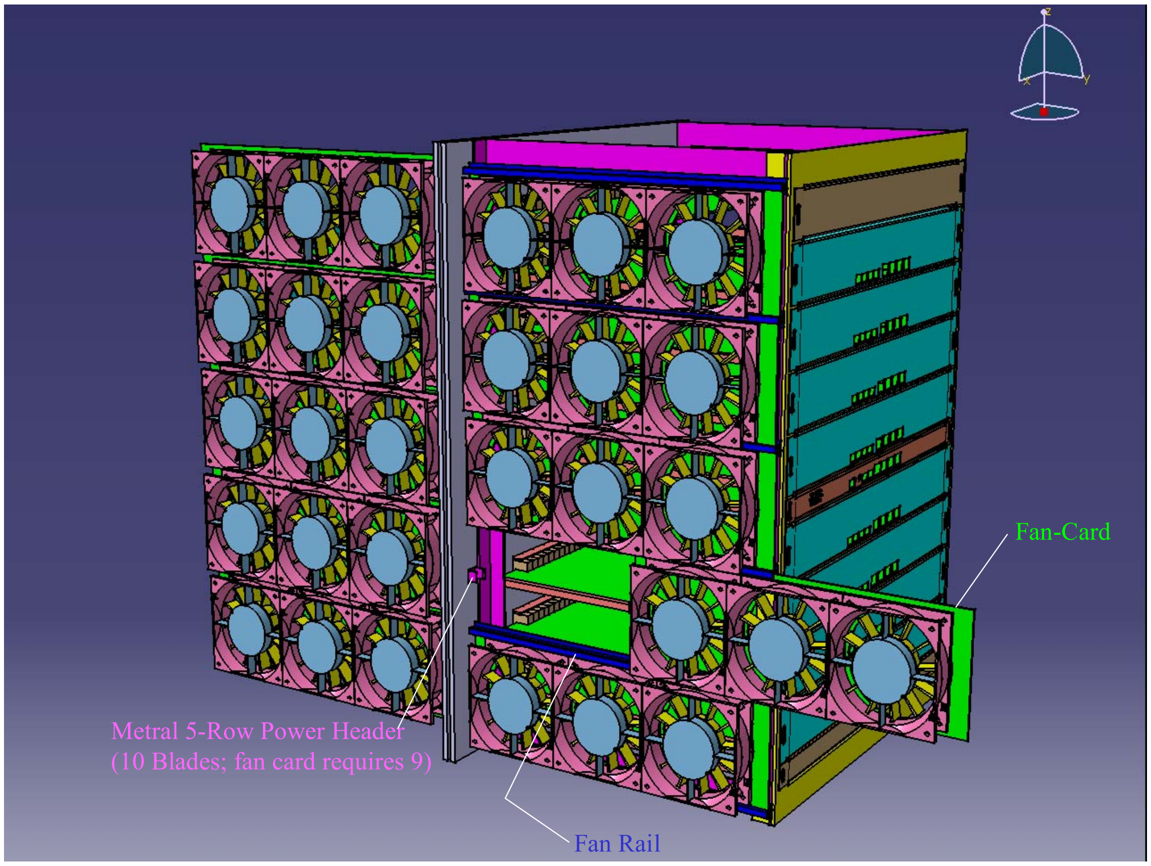

Figure 3: BlueGene/L Midplane Package.

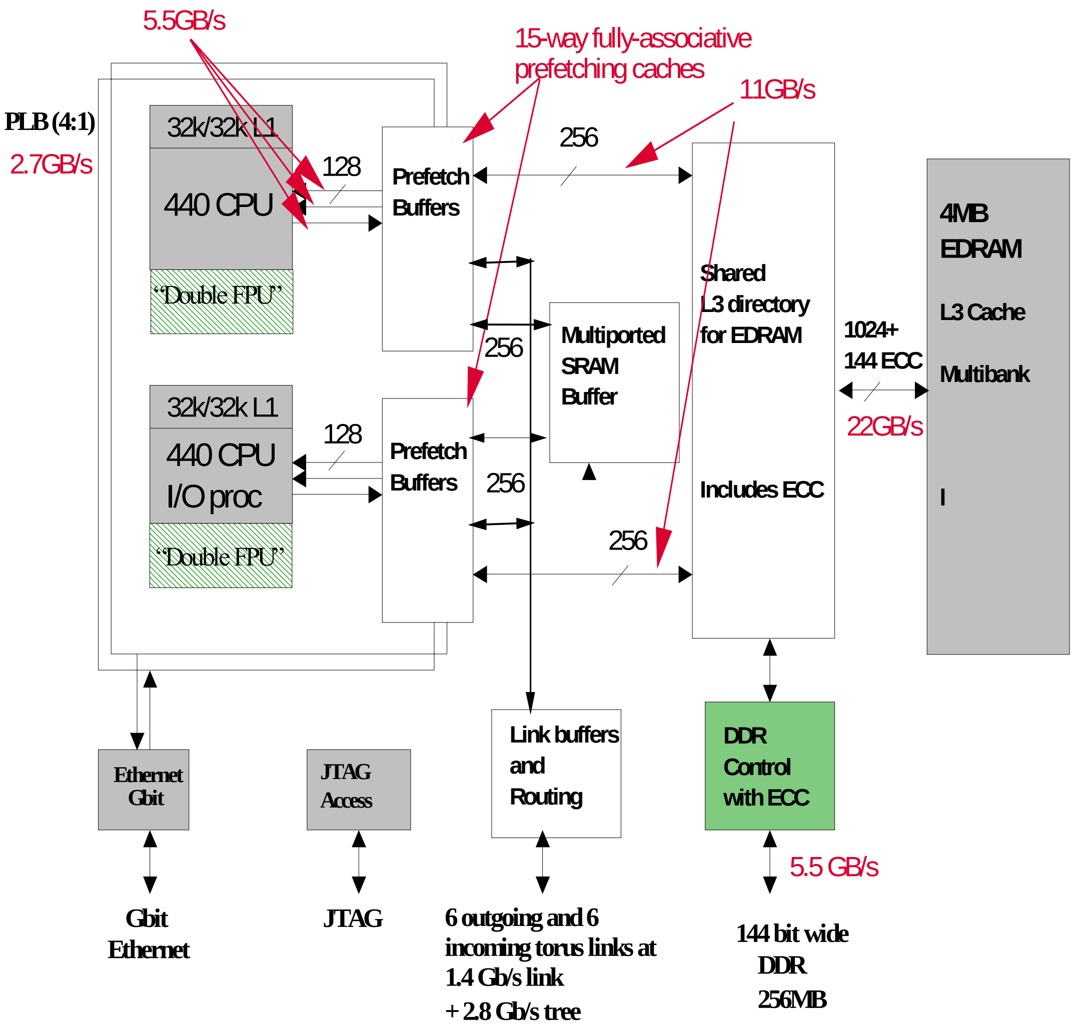

Figure 4: BlueGene/L Node Diagram. The bandwidths listed are targets.

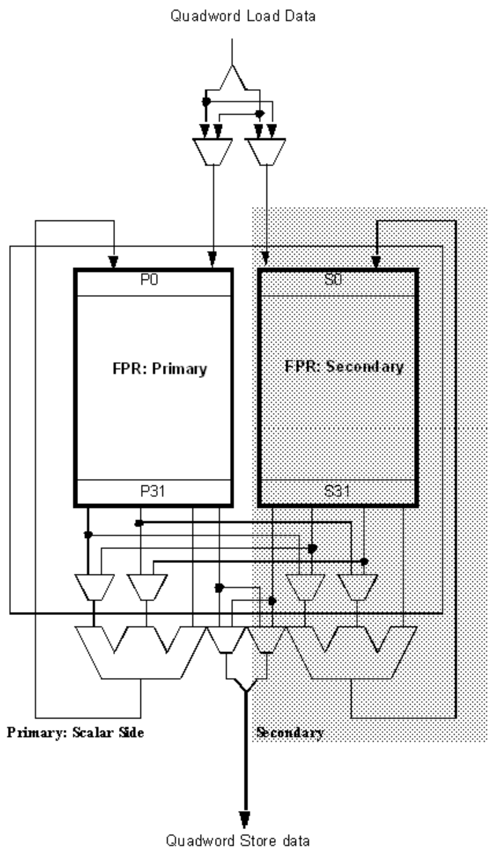

Figure 5: Double FPU Architecture

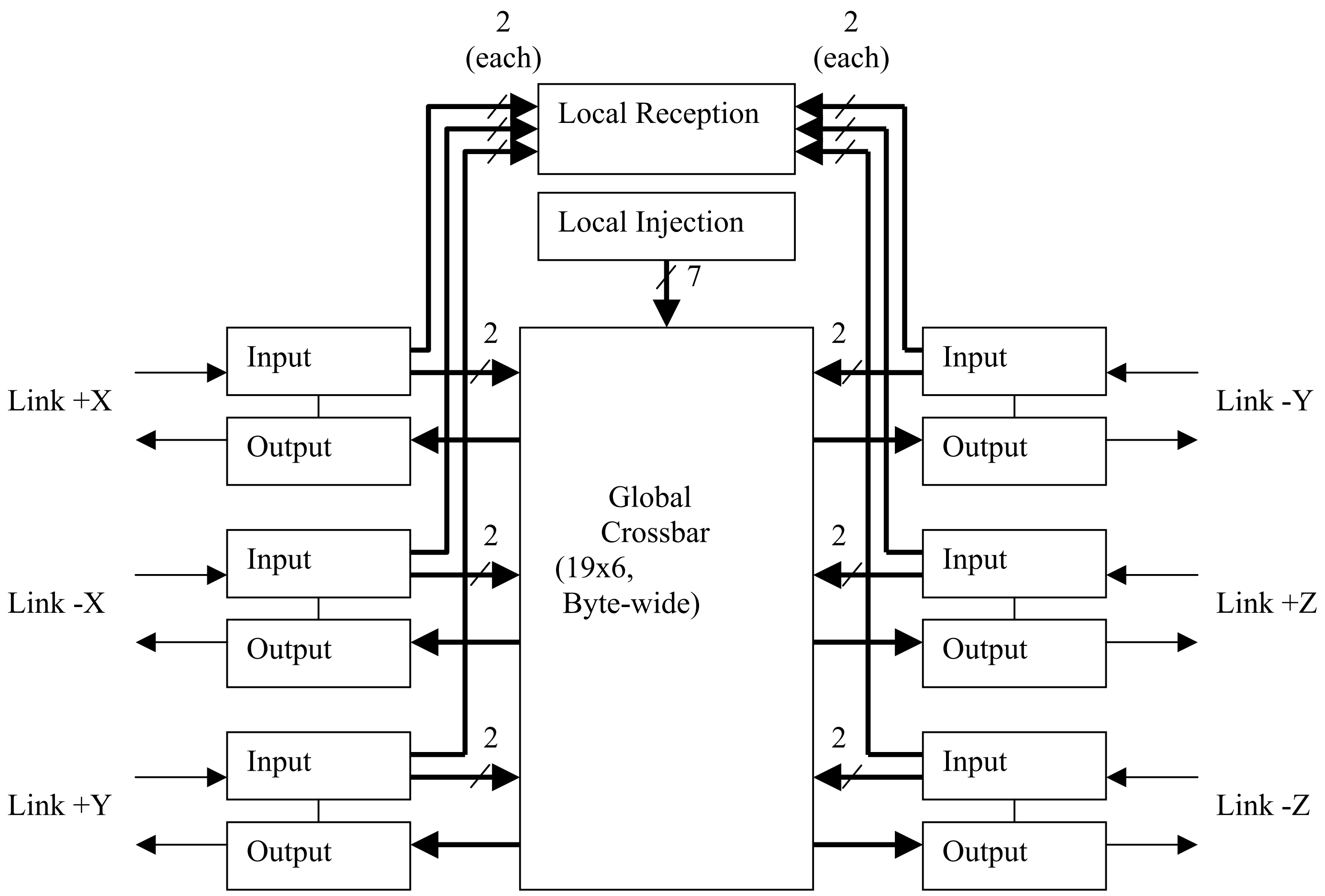

Figure 6: Basic Architecture of the Torus Router

Fact Sheet

1. 65,536 nodes available in 64 racks
2. 360 teraFLOPS peak performance goal
3. <2500 sq ft to hold 64 racks
4. 64 x 32 x 32 three-dimensional torus of compute nodes
5. Node memory: 2 GB max, 512 MB plan of record
6. Compute ASIC: CU-11 technology, 0.13 micron, 700 MHz
7. Configuration: 2 nodes/compute card, 16 compute cards/node board, 8 node boards/512-node midplane and two midplanes/rack.
8. Each processor can conduct 4 floating point operations per cycle
9. 220 Volt supply
10. 25k Watt/rack power dissipation
11. MTBF of the system is at least 10 days
12. Memory Latency: L2 hit returns in approximately 7 processor cycles, L3 hit in about 28 cycles, L3 miss in about 75 cycles
13. Torus Bandwidth: 175MB/sec in each direction
14. Global Tree: Target hardware bandwidth of 350 MB/sec and a target one-way hardware latency of about 1.5 microseconds on a 64K node partition.

Software Overview

1. O/S for Front-End Nodes and Service Nodes: SUSE SLES Version 8, Customer purchased
2. Compue Node Kernel: CNK Version 1.0, supplied with BGL Core
3. Message Passing Library: MPI Version 1.0, Customer download
4. I/O Node Kernel: Linux, distribution TBD
5. Control System: CMCS Version 1.0, supplied with BGL Core
6. Fortran Compiler: XL Fortran Version 8.1, IBM LPP (SWG)
7. C/C++ Compiler: Visual Age C++ Version 6.0, IBM LPP (SWG)
8. Data Base: DB2 Version 8.1, IBM LPP (STG)
9. Job Scheduler: Interface supplied with BGL Core to support external schedulers
10. Parallel File System: GPFS Version 2.2, IBM LPP (STG)
11. Debugger: Totalview, Customer purchased (Etnus)